

A SYSTEM AND METHOD FOR MERGING ELECTRONIC DIAGRAMS

Field of the Invention

5

The illustrative embodiment of the present invention relates generally to electronic diagrams and more particularly to merging electronic diagrams and their underlying data.

10 **Background of the Invention**

Electronic diagrams, such as flowcharts and logic state diagrams which are displayed electronically, provide an approach to explaining complex systems. Electronic diagrams are composed of labeled blocks which represent features such as system components or system logic states. Each block in the electronic diagrams has a set of properties or attributes related to the underlying represented component or logic state. The block properties may be graphical properties related to the depiction of blocks, or the properties may be functional properties related to operations of the system. The blocks may be joined by lines representing connections between the system features. The electronic diagrams may be stateless diagrams or state diagrams. A state diagram, such as a representation of a finite state machine, is an example of an event-driven system. In an event-driven system, the system transitions from one state to another provided a condition required for the change from one state to the other is satisfied (i.e.: an event of a specified kind takes place). A state represents the condition of the system at a particular point in time. A stateless diagram depicts the flow of data or operations in a system from one component to another without depicting the state of the system. The depicted system may also be a purely computational system.

Both stateless diagrams and logic state diagrams are often depicted in an electronic format through the use of a computer diagramming application, such as Simulink™ running on MATLAB™, from The MathWorks Inc. of Natick, Massachusetts. The diagramming applications provide displays of electronic diagrams to users of an electronic device. Conventionally, the electronic version of a diagram is stored on an electronic device, such as a computer system, which is interfaced with a

display device that presents the electronic diagram to a user. Techniques available in graphical diagramming applications enable multiple diagrams to be presented to a user in a side-by-side display. Unfortunately, there currently exists no easy method of merging differences between two electronic diagrams being displayed to a user.

5

Brief Summary of the Invention

The illustrative embodiment of the present invention provides a method of reconciling and merging two displayed electronic diagrams into one electronic diagram.

10 After differences between corresponding areas of the two electronic diagrams are detected, the method of the illustrative embodiment of the present invention provides a mechanism for merging different features of a first of the diagrams into a second of the diagrams. The second diagram may receive and merge all of the differences identified from the first diagram, or only selected differences. Distinctions are made between
15 graphical and functional feature differences in the two diagrams. The illustrative embodiment of the present invention enables the user to specify which type of feature differences should be merged (i.e., graphical differences, functional differences, both graphical and functional differences, or individually selected differences regardless of the classification).

20

In one embodiment of the present invention, an electronic device that is interfaced with a display device displays two electronic diagrams. Corresponding sections of the two diagrams are identified and the differences, if any, between the two diagrams are recorded. Selected differences from the first diagram are merged into the
25 second diagram at the corresponding section in the second diagram for each of the differences.

In another embodiment of the present invention, an electronic device that is interfaced with a display device displays two state diagrams. Corresponding sections of
30 the two state diagrams are identified and the differences, if any, between the two state diagrams are recorded. Selected differences from the first state diagram are merged into

the second state diagram at the corresponding section in the second diagram for each of the differences.

In an additional embodiment of the present invention, an electronic device is
5 interfaced with a network and with a display surface. The display device is used to
display two electronic diagrams which are retrieved from the network to the electronic
device. Corresponding sections of the two electronic diagrams are identified and the
differences, if any, between the two electronic diagrams are recorded. The user-selected
differences from the first electronic diagram are merged into the second electronic
10 diagram at the corresponding section in the second diagram for each of the differences.

Brief Description of the Drawings

Figure 1 depicts an environment suitable for practicing the illustrative
15 embodiment of the present invention;

Figure 2 depicts a top level view of the sequence of steps followed by the
illustrative embodiment of the present invention to merge differences from one diagram
to another;

Figure 3A depicts a block diagram of displayed differences in the illustrative
20 embodiment of the present invention;

Figure 3B depicts a block diagram of a report of the identified differences
between two electronic diagrams in the illustrative embodiment of the present invention;

Figure 4 depicts the sequence of steps followed by the illustrative embodiment
of the present invention in performing an exact merge; and

25 **Figure 5** depicts a sequence of steps followed by an illustrative embodiment of
the present invention in performing an heuristic merge.

Detailed Description

30 The illustrative embodiment of the present invention provides a method for
merging two electronically displayed diagrams, such as flowcharts or logic state
diagrams. The diagrams are analyzed to determine points of correspondence (i.e.: the

points where the two diagrams are alike). The system components of the two diagrams are examined for differences. Differences between corresponding sections of the two diagrams are recorded as difference items. The difference items are further divided into functional differences and cosmetic differences. After presentment of the differences to
5 a user of an electronic device, the selected differences for some or all of one of the diagrams are merged into corresponding sections of the other diagram.

Figure 1 depicts an environment suitable for practicing an illustrative embodiment of the present invention. An electronic device 2 includes memory 4. The
10 memory 4 holds a diagramming application 6 capable of creating electronic versions of system diagrams such as block diagrams, state diagrams, signal diagrams, flow chart diagrams, sequence diagrams, UML diagrams, dataflow diagrams, circuit diagrams, ladder logic diagrams or kinematic element diagrams. A suitable diagramming application 6 is an application such as MATLAB, version 6.1 with Simulink, version
15 4.1. Also held in memory 4 is a differencing engine 7 which is used to detect differences between electronic diagrams. A display device 8 is interfaced with the electronic device 2 enabling the display of two diagrams from the diagramming application 6 to a user 10. In an alternate embodiment, the electronic device 2 is also interfaced with a network, such as the Internet. Those skilled in the art will recognize
20 that the diagrams used by the diagramming application 6 may be stored either locally on the electronic device 2 or at a remote location interfaced with the electronic device over a network. Similarly, the diagramming application 6 may be stored on a networked server or a remote peer. The illustrative embodiment of the present invention analyzes the differences in system components in two diagrams, presents them to a user 10, and
25 then merges the user-selected differences from one diagram into another. If the outcome of the merge operation is acceptable to the user 10, the user may save the outcome of the operation which merges the data in the underlying model or simulation that is being displayed as electronic diagrams. If the outcome of the merge operation is not acceptable to the user 10, the outcome is not saved and the underlying data is
30 undisturbed by the merger operation.

The data represented by the electronic diagrams may be stored in any one of a number of different types of data structures. In one embodiment of the present invention, the diagrams use Simulink data objects to store the diagram's data. The simulink data objects may be used to define MATLAB data types that are specific to a user's application. Simulink includes built-in blocks which may be used to implement modeling functions desired by a user. The blocks in the electronic diagrams may model the behavior of specialized mechanical, circuit or software components, such as motors, servo-valves, power plants, filters, tires, modems, receivers and other dynamic components. The data objects hold both block attributes and functions. Those skilled in the art will recognize that the functions may include code written in C, Ada, Fortran, MATLAB or other programming languages. The electronic diagrams may encompass multiple domains within a single diagram.

The illustrative embodiment of the present invention includes a differencing engine 7 which walks down the hierarchy of the electronic diagram and finds blocks and connections that are present in both models or in only one model and compares them for differences. The differencing engine 7 first compares the two diagrams to determine points of correspondence between the two diagrams. This is accomplished using a matching algorithm. The matching algorithm applies rules to match components in the two diagrams. In one embodiment, the matching algorithm rules dictate that blocks or subsystems with different names are considered unique, even if everything inside of them is the same. In another embodiment, the matching algorithm rules dictate blocks or subsystems with the same positions or some pre-defined set of properties are considered to be a match, even if the names are different. Connections between blocks are considered unique if the source block or destination block or source port or destination port does not exist in both models. A port may be either a physical connection into which a device connects, or an endpoint to a logical connection (i.e.: port 80 is used for HTTP traffic in a TCP/IP network), depending upon the type of system being modeled by the electronic diagram. Ports of a block which are connected in one model and are not connected in the other model are also considered to be unique differences. In one embodiment, lines which are not connected to a source block and a destination block, are not compared at all for differences. In another embodiment, lines

which are not connected to a source block and a destination block are compared for differences. The results of the matching algorithm are stored in a dataset containing a list of related pairs of corresponding electronic diagram elements, one from diagram A and one from diagram B in each entry in the list (where A and B are the two electronic diagrams being compared). Each item in the list that is unique or different is a difference item. The illustrative embodiment of the present invention is not dependent upon the use of any one particular method of detecting differences between electronic diagrams, as long as the detected differences are made available to the diagramming application 6 for merge operations.

10

In one embodiment of the present invention, the data structure used by the differencing engine 7 to record and track differences between two diagrams is a $N \times 1$ MATLAB cell array (where N is an integer value). A cell array is a MATLAB array for which the elements are cells. The cells are containers that can hold other MATLAB arrays, such as a real matrix, an image, a structure, an array of text strings, or a vector of complex values. Each cell in the cell array contains a cell array of 8 items. A unique handle, a floating point number, is assigned to each electronic diagram data element being stored in the cell array. Data items that are found to be different are marked as eligible for merge operations. Data items containing a child element are also marked for hierarchical operations. Those skilled in the art will recognize that the type of data structure the differences are stored in may be altered without departing from the scope of the present invention.

Once matches have been established by the matching algorithm in the differencing engine 7, a compare algorithm compares each matched item for properties. The set of properties may be programmatically defined or chosen by a user. In one embodiment of the present invention, properties are broken down into two groups, graphical properties and functional properties. Graphical properties affect the appearance of the diagram, functional properties affect the performance of the depicted system. For example, a graphical difference occurs when two blocks, a block A and a block B, appear in both diagrams, but the first diagram locates the blocks in a different position in the diagram. A functional difference occurs when the two blocks produce

30

different results in each diagram. Blocks may be considered to be different by the differencing engine 7 if any property is different. Connections may be semantic connections or direct connections. Semantic connections include GOTO and From blocks. Connections may be considered to be different when both the blocks and ports, source and destination, exist in both models but they are connected to a different block or port than that in the other model. Once the differences have been determined, the diagrams may be displayed to the user 10. In one embodiment of the present invention, diagram elements are listed side by side in a tree format, with the differences highlighted on the display device 8.

10

Figure 2 is a flow chart of the sequence of steps utilized by the illustrative embodiment of the present invention to merge differences between two electronic diagrams. The sequence begins when the differencing engine 7 detects differences in corresponding sections of two diagrams, such as flowcharts or state diagrams (step 20). The differences are highlighted and presented to the user 10 on the display device 8 (step 22). The user 10 selects the type of differences subject to the merge process and the direction in which the merging process should take place(i.e.: from the first diagram to the second diagram, or from the second diagram to the first diagram) (step 24). In an alternate embodiment, the type of difference being merged is pre-programmed in the differencing engine 7. The user 10 may also select specific corresponding sections of the two diagrams subject to the merge process (step 26). The diagramming application 6 performs the merging process selected by the user (step 28). Once complete, the two diagrams are presented to the user with the selected diagram having undergone the merge process (step 30). The user has the ability to activate an Undo feature to restore the electronic diagrams to their pre-merge status if dissatisfied with the result of the merge operation. Those skilled in the art will recognize that the merge process may have involved the entire diagram or only selected sections or attributes of the electronic diagram.

30 Merging commands in the illustrative embodiment of the present invention are processed by taking the output of the differencing engine 7 as input to the merge software which edits the electronic diagram per the user command. In the illustrative

embodiment, the merging software emits diagram editing instructions to the diagramming software 6 via the diagramming software's editing API as a means of manipulating the destination diagram using the selected difference region data element(s) from the differencing engine 7 output. For example, if the current difference
5 region is a block that exists in both diagrams A and B and the block property "SampleTime" differs between diagrams A and B and the user commands that the "SampleTime" property of the block in diagram A is to be merged to the matching block in diagram B, the merging software will match the current region's block handle with a unique handle entry in the current difference region in the differencing engine 7 output
10 and retrieve the corresponding index of the cell element containing the difference region. The different property of the current difference region, "SampleTime" is changed using the diagramming software's parameter value editing function, `set_param(h, 'SampleTime', x)`, where x is the value of the "SampleTime" property in diagram A and h is the handle of the block in diagram B.

15

In the illustrative embodiment, Simulink block diagrams are edited using APIs such as `add_block`, `add_line`, `delete_block`, `delete_line`, `get_param` and `set_param`. Stateflow diagrams are edited with APIs such as `sf('get', ...)` and `sf('set', ...)`. For example, to change the property SampleTime of a Stateflow object h to value x, the
20 merge software will use `sf('set', h, 'SampleTime', x)`. To retrieve the value of "SampleTime" from Stateflow object h, the merge software will use `sf('get', h, 'SampleTime')`. The particular means or API used to edit the electronic diagram is not critical to the merge operation as long as the merge software can manipulate the electronic diagram sufficiently to complete the merge operation. The merge software
25 may be part of the diagramming application 6 or may be a separate application.

When the diagramming application 6 first begins the merge process in one embodiment of the present invention, the user 10 is presented with four GUI components (the GUI components are illustrated in Figure 3A below). The display
30 device 8 displays all of the graphical difference items found in electronic diagrams A and B including any number of graphical child data elements. Each of the child data elements includes any number of property data elements. The user 10 is also presented

with a display of a first diagram with a “current” difference item highlighted. A “current” difference item is one of the identified difference items in the diagram which is visually accentuated to focus the user’s attention. The user may replace the current difference item with a different difference item (which then becomes the current
5 difference item). The third GUI component is a display of a second diagram with a current difference item highlighted. The electronic diagram’s internal data format has no relevance to the merging operation. A fourth GUI component, an attribute difference panel is used to present an itemization of differences in data objects in the two electronic diagrams.

10

Figure 3A depicts the GUI components that are used to present identified differences between two electronic diagrams to the user 10. The display device 8 presents three windows 31, 32 and 33 to a user. An electronic diagram entitled “one” is presented to the user 10 in a window 31 in the top right of the display surface of the
15 display device 8. An electronic diagram entitled “two” is presented to the user 10 in a window 32 in the bottom right of the display surface of the display device 8. A third window 33 (“the difference window “) holding identified differences (optionally together with all diagram elements for context) between the two electronic diagrams is presented to the user 10 on the left side of the display surface of the display device 8.
20 The difference window 33 includes two trees 34 and 35 listing the identified data objects listed side by side so that corresponding locations in the two electronic diagrams appear next to each other. Thus, both trees 34 and 35 list a block A and a block B and a first connection that runs between the two blocks in the same order. A second connection 36 running between the two blocks appears only in the first electronic diagram and
25 therefore appears only in the tree 34 for electronic diagram “one”. The second connection 36 is a unique item and is therefore highlighted in a manner designed to bring it to the user’s attention. A fourth window 37 presents an itemization of data object differences, indicating that blocks “one” and “two” have a different number of ports. Those skilled in the art will recognize that the positions of the windows on the
30 display surface of the display device 8 are an implementation choice which may be altered without departing from the scope of the present invention. Similarly a different data presentation format may be substituted for the tree structure discussed above.

As noted above, the results of the differencing operation are stored in a dataset. The dataset lists the components in the two diagrams that are involved in the diagram and the type of object that appears as a difference. In addition to being graphically
5 presented to the user 10 as outlined in **Figure 3A** the differences contained in the dataset may also be presented to the user 10 in a report. **Figure 3B** depicts the presentation of a static difference report 38 to the user 10. The report 38 may include hyperlinks back to the electronic diagrams. The report 38 includes a column 39 for the object type of the difference identified, and columns 40 and 41 for the involved objects in the two
10 electronic diagrams. The report 38 indicates that block A and B in electronic diagram “one” and “two” have differences within the blocks.

The illustrative embodiment of the present invention enables a user to specify which of the difference items within a selected section of a diagram to merge into the
15 other diagram. An “exact merge” copies all of the difference items in a selected diagram section from one diagram to the other. **Figure 4** depicts the sequence of steps followed by the illustrative embodiment of the present invention in performing an exact merge. First, corresponding areas of two displayed diagrams are established by the matching algorithm in the diagramming application 6 (step 42). The corresponding
20 sections of the two diagrams contain system components and connections which are present in both diagrams. After corresponding sections of the two diagrams have been determined, the diagramming application 6 uses a compare algorithm to establish difference items between the corresponding sections of the diagrams (i.e., attributes which are different from one diagram to the other) (step 44). Difference items are
25 highlighted and presented to the user 10 who then selects which difference items to merge. This selection also involves the user 10 determining in which direction the merge process is to run (i.e.: whether the differences should be merged from diagram A to diagram B, or alternatively, from diagram B to diagram A). Once selected, the difference items are then copied (step 46).

30

The different data elements making up the system components or states in the diagrams may include sublevels associated with the represented components. For

example, a parent component may have multiple sub-components arranged in a parent and child tree-like structure. The illustrative embodiment of the present invention provides a method of merging both parent and child elements which may be selected by the user 10. A determination is made as to whether or not the corresponding section of the destination diagram is “hierarchical”, that is, whether or not the destination data element has child elements associated with the displayed parent element (step 48). If the destination data element is not a hierarchical data element, then only the destination data element is replaced by the copied difference item (step 50). Alternatively, if the destination data item is a hierarchical item possessing child elements, the destination data element and all of the child elements are replaced by the copied difference item (step 52). The user may also choose to copy only portions of a hierarchical difference item such that only a parent portion of a data element is merged and replaced, or alternatively, only a child or several generations of children elements are merged and replaced while the parent element is left untouched. Once the merge process has been completed, the results of the merge are presented to the user 10 on the display device 8 (step 54).

The illustrative embodiment of the present invention categorizes detected differences according to type. Some of the differences identified in the two displayed diagrams are the result of functional differences in system components or system states. Other differences on the displayed diagrams are the result of graphical or cosmetic differences in the manner in which the system diagram is presented to the user. The illustrative embodiment provides a method of selecting which types of differences to merge from one diagram to the other. The user may select a merger of all of the functional differences and those cosmetic differences not requiring major revisions to the destination diagram. The determination as to whether the revision would be a major revision or not is based on pre-defined parameters regarding how much of the drawing would have to be re-drawn. Examples of cosmetic or graphic differences are the position of a block in a subsystem or the path of a line from one block to another in a subsystem.

Mergers where all of the functional differences and less than all of the cosmetic or graphical differences are copied are referred to “heuristic”. **Figure 5** depicts the sequence of steps followed by the illustrative embodiment of the present invention in performing a heuristic merge. A heuristic merge begins when the matching algorithm in the diagramming application 6 establishes corresponding areas between the two diagrams (step 60). Once the corresponding areas of the two diagrams have been established, differences within the corresponding areas are determined and difference items determined using the compare algorithm of the diagramming application 6 (step 62). The differences are categorized as either functional or cosmetic/graphical. All of the functional differences in the difference items selected items are copied (step 64). Less than all of the graphic differences are also copied (step 66). The selected differences are then merged with the destination diagram thereby replacing the corresponding data elements in the second diagram with the copied difference items (step 68). Once the merge operation is complete, the results are presented to the user (step 70). Those skilled in the art will recognize that the heuristic merge is used when merging difference items that have graphical incompatibilities in the destination electronic diagram such as connections between blocks of different positions, yet a line connection between the blocks is still desired by the user 10. Similar to the exact merge, the heuristic merge may also cascade hierarchically. If the heuristic merge is utilized, it replaces all of the parent and child data elements as noted above.

The illustrated embodiment of the present invention also enables the user 10 to perform a merge of one of the current difference items. The merge of the current difference item is referred to as a “contextual merge”. Using either GUI buttons, keyboard mappings or menu items, the user 10 may command an immediate merge of either of the current difference items to replace the other current difference item, either A to B or B to A. If an exact merge is possible, it is performed. If an exact merge is not possible a heuristic merge is performed. If the data elements of the difference item being merged contains child data elements, the merge is hierarchichally cascaded.

The illustrative embodiment of the present invention allows a user to programmatically merge two displayed diagrams. By selecting the type of merge

operation, the user may control the features that are copied from one diagram to another. By selecting merge locations, the user can control which parts of the diagrams are merged. Additionally, the user may select specific features from within an area to merge and the merge operation may be performed in both directions within a single operating session. The ability to perform the merge operations programmatically enables multiple versions of similar diagrams to be rapidly compared.

The API of the differencing engine 7 can be utilized to extract differences between an electronic diagram in memory and an electronic diagram stored in a configuration management system (a/k/a “revision control system “) by using a single command executed in the MATLAB language environment. For example, three electronic diagrams A, B, C, may be compared separately by the differencing engine 7 to a single electronic diagram X stored in a configuration management system. The differences between electronic diagrams A, B and C in memory and the single electronic diagram X stored in the configuration management system can be applied sequentially to X (to the degree the difference regions do not intersect) to merge all differences from A, B, and C into X and store the merged X as a new revision of X in the configuration management system. As long as the differences in A, B, and C with respect to X do not intersect with each other, the order of differencing of A, B, and C with respect to X can occur in any order. Merging can begin after differencing is complete.

In one embodiment two electronic diagrams are provided that have blocks representing components of a system. Corresponding features of the electronic diagrams are determined that are present in both of the electronic diagrams. Differences between the electronic diagrams are determined and the differences are copied from a selected one of the electronic diagrams and merged into the other electronic diagram at a corresponding location in the other electronic diagram. The differences in the electronic diagrams are highlighted for a user on a display surface of a display device. The display surface shows both of the diagrams and is updated following the performance of the merging operation. The updating shows the differences copied to the other electronic diagram. In one embodiment, a distance is determined on the display surface from an endpoint of a line to an updated connection point for a block in said electronic diagram. The updated connection point is the connection point of a line and a block following a

merge operation. The determined distance is compared to a pre-defined parameter that is a distance value. The displayed line is extended to the updated connection point when the distance is less than the pre-defined parameter. In an embodiment, the displayed line is replaced with a new line drawn to the updated connection point when the distance is
5 at least as large as the pre-defined parameter.

In an embodiment, two state diagrams of a system are provided. The state diagrams have blocks joined with lines with each of the blocks representing states in the system. The lines represent transitions between the states that take place upon the
10 occurrence of a specified event. Corresponding features of the state diagrams are determined that are present in both of the state diagrams. Differences between the state diagrams are determined and are recorded as a list of data elements. The differences are merged from a selected one of the state diagrams into the other of the state diagrams. The merging copies the differences from the selected one of the state diagrams and
15 inserts the differences in the other state diagram.

It will thus be seen that the invention attains the objects made apparent from the preceding description. Since certain changes may be made without departing from the scope of the present invention, it is intended that all matter contained in the above
20 description or shown in the accompanying drawings be interpreted as illustrative and not in a literal sense. Practitioners of the art will realize that the system configurations depicted and described herein are examples of multiple possible system configurations that fall within the scope of the current invention. Likewise, the sequence of steps utilized in the illustrated flowcharts are examples and not the exclusive sequence of
25 steps possible within the scope of the present invention.